



ILab Parameter Study and Globus Auto-submission Tool

M. Yarrow and K. McCann

with

R. Biswas and R. Van der Wijngaart

NASA Ames Research Center



◆ Objective

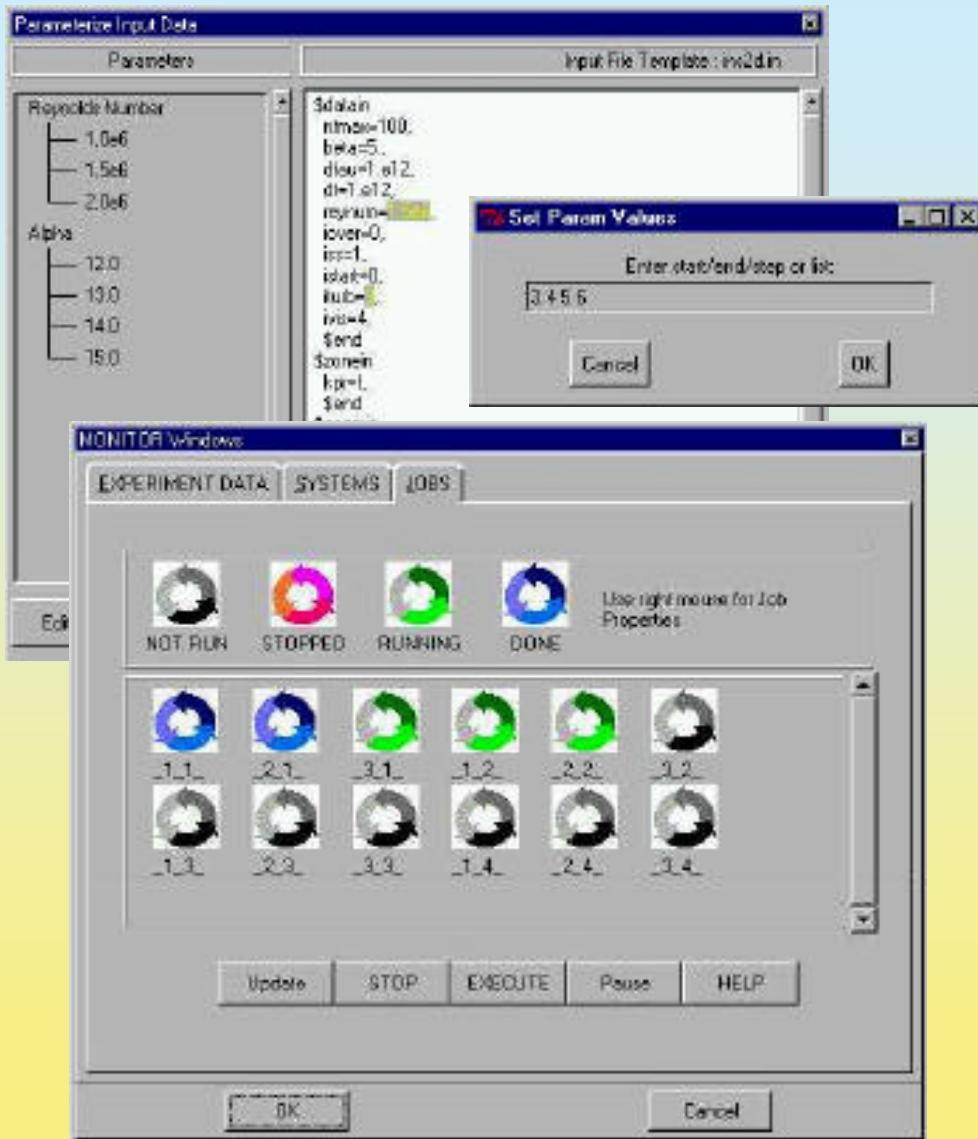
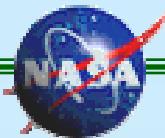
- User-friendly tool for Parameter Study applications on IPG

◆ Requirements

- Automated high throughput
- Globus enabled
- Monitoring, logging, and archiving
- Ease of use
- Handle complex multistage processes

◆ Relevance to IPG and NASA

- Automates important functionality (which is currently performed manually)
- Secretarial side of Problem Solving Environment
- Reduces design cycle time



ILab : The Information Power Grid Virtual Laboratory

Parameterization made simple
and easy

Generates and submits scripts to
execute parameterized jobs

Absolutely no programming or
scripting required

Several Job Models, including Globus,
PBS, MPI, local or remote

Real-time monitoring of job status

Organization and archiving of
all "Experiment" data

MRU and complete "history" secretarial
features

Built with Object-Oriented
programming model

FUTURE :

CAD-based complex process
specification;

Additional Job Models : Condor,
Legion, PVM



PROGRAMMING CONSIDERATIONS

Object-Oriented Programming model :

- uses Perl “package” which is equivalent to C++/Java “class”
- objects are nested 4 deep :
 - Experiment -> Process -> ParamFile -> ParamData
- uses CONTAINMENT, not INHERITANCE
 - (- as recommended by OOP experts)
 - (- although, some inheritance in Tk dialog objects)

Experiment object is dumped to/restored from DISK with

MRU (“Most Recently Used”) feature : one mouse click restores
ALL data associated with an Experiment

ON-LINE HELP for user convenience: HELP in main window, plus a
HELP button in every screen and dialog

ILab now contains about 18 screens and pop-up dialogs : more will
be added.

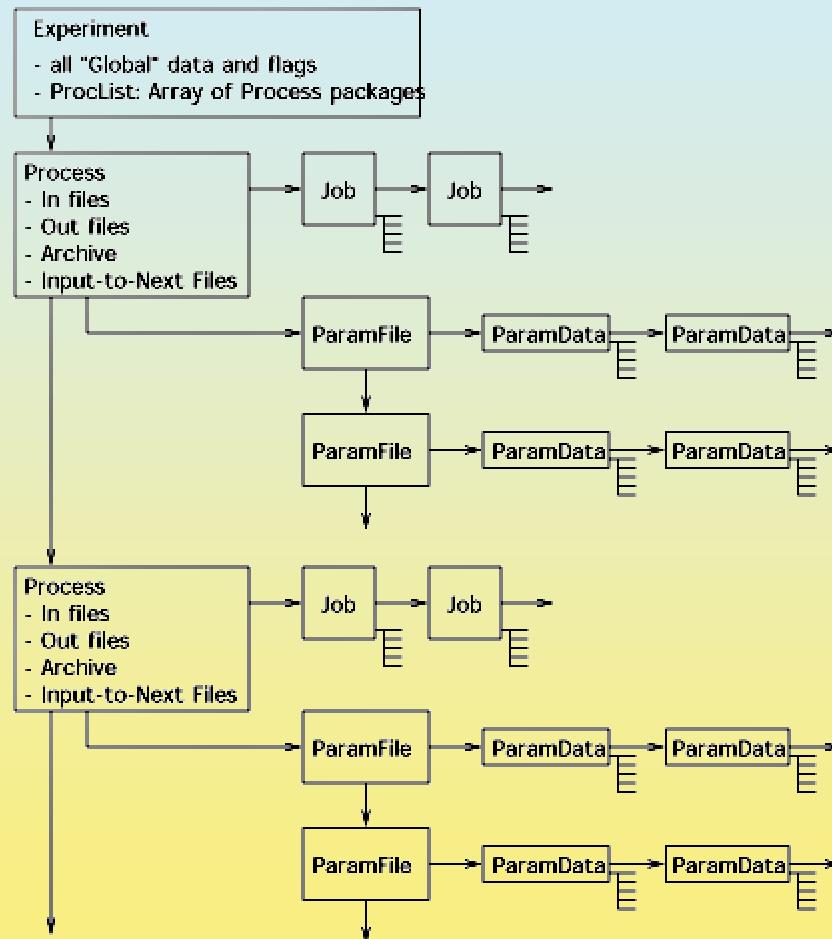
FAST Rapid Prototyping : ~14,000 lines of Perl in 4 months!

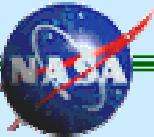
- used “SpecTcl” to generate code for dialogs and windows
(primitive IDE available from Sun Microsystems)

SYSTEMS : ILab GUI runs on both UNIX and PC



ILab OBJECT CONTAINMENT HIERARCHY





EXAMPLE OF AUTOMATICALLY GENERATED KSH SCRIPT

```
#!/usr/bin/ksh

#
# All attributes are set in this block
#
unixbin=/usr/bin
bsdbin=/usr/bsd
rcp_cmd=rcp
exec_rootdir=/u/wk/yarrow/lLab_ins2d_test17
param_file_suffix=_1_2_
next_param_file_suffix=_2_2_
param_fname=ins2d.in
param_deck_tempdir=/u/wk/yarrow/lLab_ins2d_test17/temp_0009181124

#
# Check for existence of $exec_rootdir; if it does not exist, create one
#
echo $param_file_suffix: checking for $exec_rootdir
if [[ -e $exec_rootdir && ! -d $exec_rootdir ]]
then
    echo $param_file_suffix: $exec_rootdir exists, but is not a directory
    echo am exiting...
    exit
fi
if [[ ! -d $exec_rootdir ]]
then
    echo $param_file_suffix: making dir $exec_rootdir
    $unixbin/mkdir -p $exec_rootdir
fi

#
# Create $exec_rootdir/$param_file_suffix subdirectory
#
# Note: what happens if this below dir already exists?
if [[ ! -e $exec_rootdir/$param_file_suffix ]]
then
    echo $param_file_suffix: making dir $exec_rootdir/$param_file_suffix
    $unixbin/mkdir $exec_rootdir/$param_file_suffix
fi

#
# Go to run directory
#
cd $exec_rootdir/$param_file_suffix

#
# Set the names of files to be copied:
#
InFilePaths_1="/u/wk/yarrow/ins2d2.2b/sample/airfoil/*"
InFilePaths_2="/u/wk/yarrow/ins2d2.2b/sample/airfoil/ins2d.in"

#
# Copy $InFilePaths to run dir
#
$unixbin/cp $InFilePaths_1 .
$unixbin/cp $InFilePaths_2 .

#
# copy the param file to rootdir
#
echo $param_file_suffix: rcp param deck
$unixbin/cp -p $param_deck_tempdir/$param_fname.$param_file_suffix $param_fname

#
# Set the executable program command line
#
execution_string="/u/wk/yarrow/ins2d2.2b/sample/airfoil/ins2d < ins2d.in > ins2d.out"

#
# Invoke the above execution_string
#
echo $param_file_suffix: running job 1: ins2d
eval $execution_string

#
# Set the executable program command line
#
execution_string="/u/wk/yarrow/ins2d2.2b/sample/airfoil/postproc1.pl "
```



STEPS IN CREATING AND SUBMITTING PARAMETER STUDY

- 1) Specify names for “Experiment”, remote super-directory, and remote machines
- 2) Specify input files
- 3) Parameterize input file with built-in parameterizer
- 4) Specify Process data, executable name
 - resource requirements
 - scheduler, if any (currently PBS supported, LSF coming)
 - metacomputing environment, if any (currently Globus supported)
 - parallel setting and launcher, if any (mpirun currently supported)
- 5) Specify output files to be archived, if any
- 6) Move to next process, if any
- 7) Submit and monitor

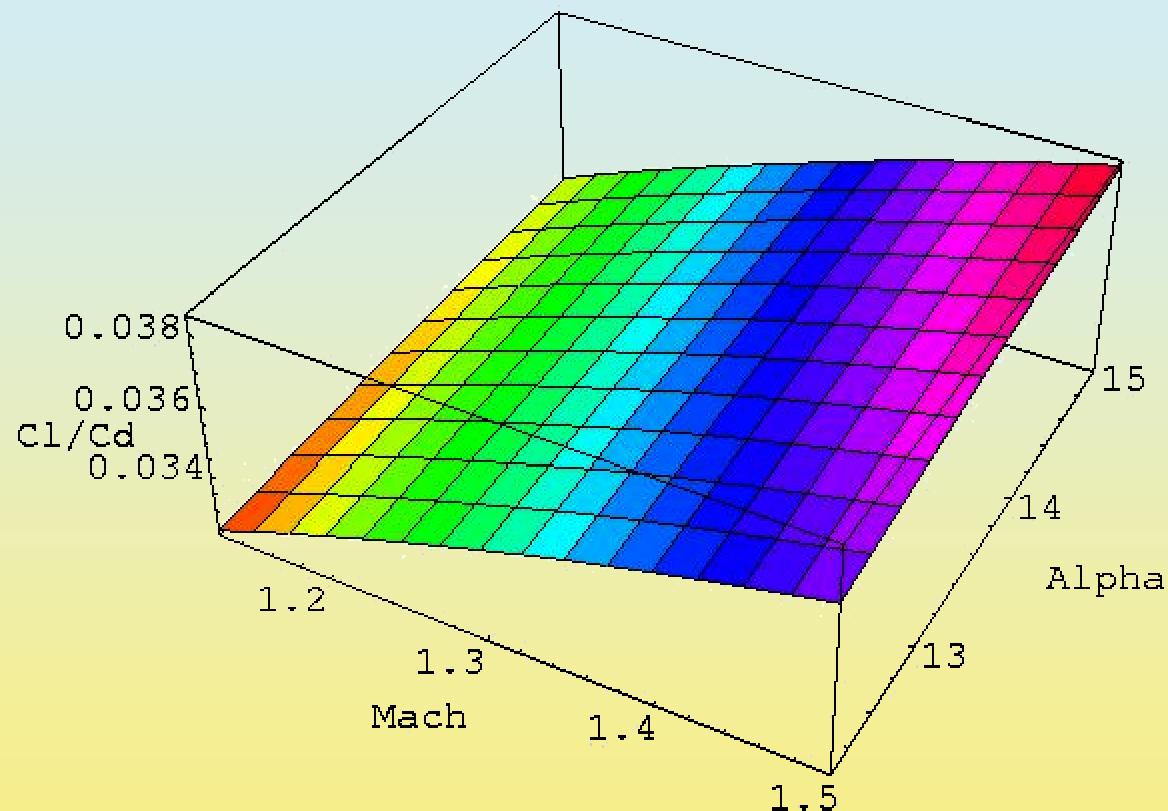


WHAT ILAB DOES - GLOBUS JOB MODEL

- 1) Generates parameterized input files
- 2) Generates shell scripts
 - local scripts
 - remote scripts: contain embedded automatically generated Globus RSL string, scheduler control language, appropriate shell scripting
- 3) Submits local script
 - selects target machine
 - migrates and submits remote scripts via Globus
- 4) Remote script (one for each job)
 - creates subdirectory for run
 - pulls required files and executables from source location
 - runs executables
 - archive output if requested

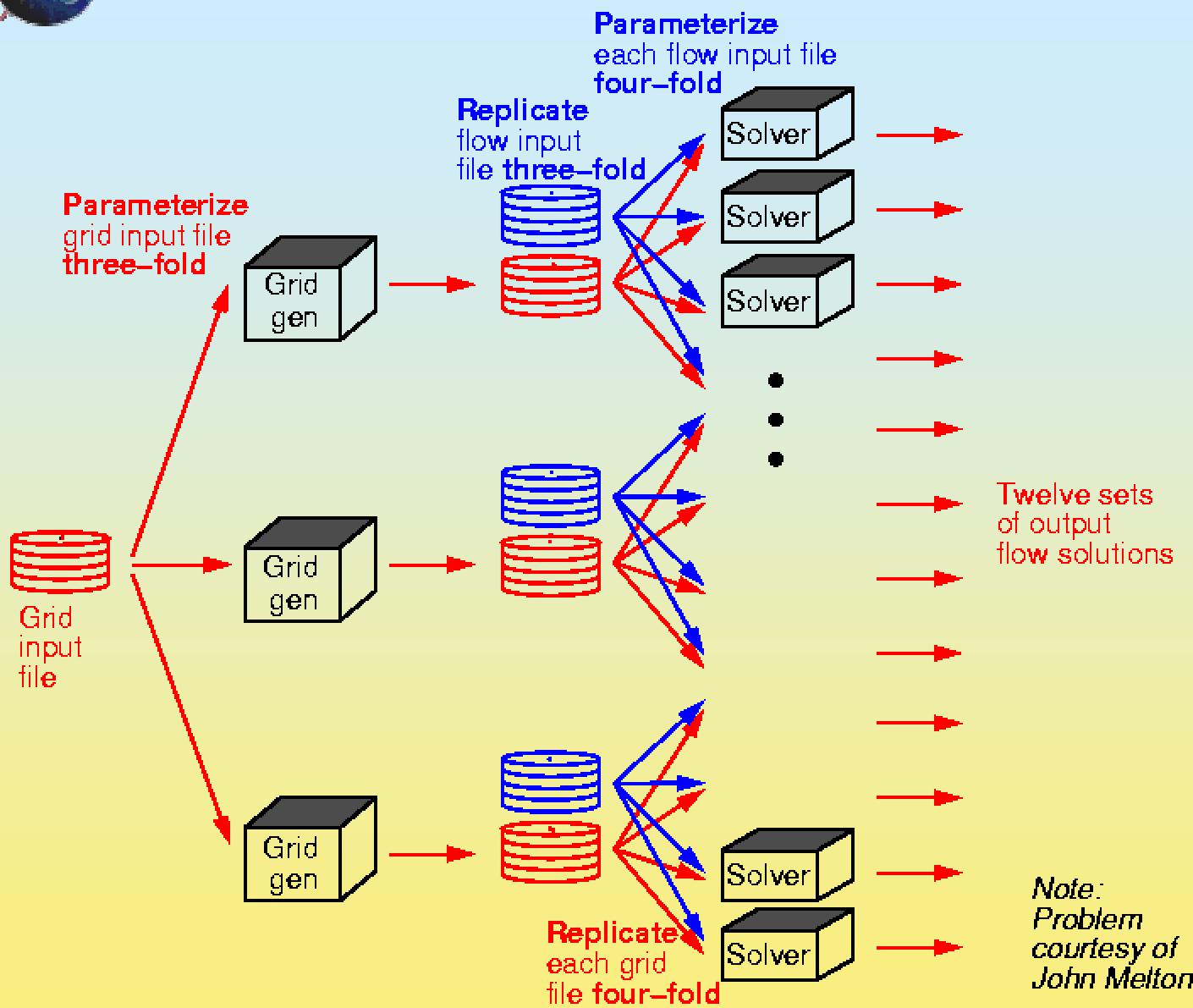


COEFFICIENT OF LIFT OVER DRAG



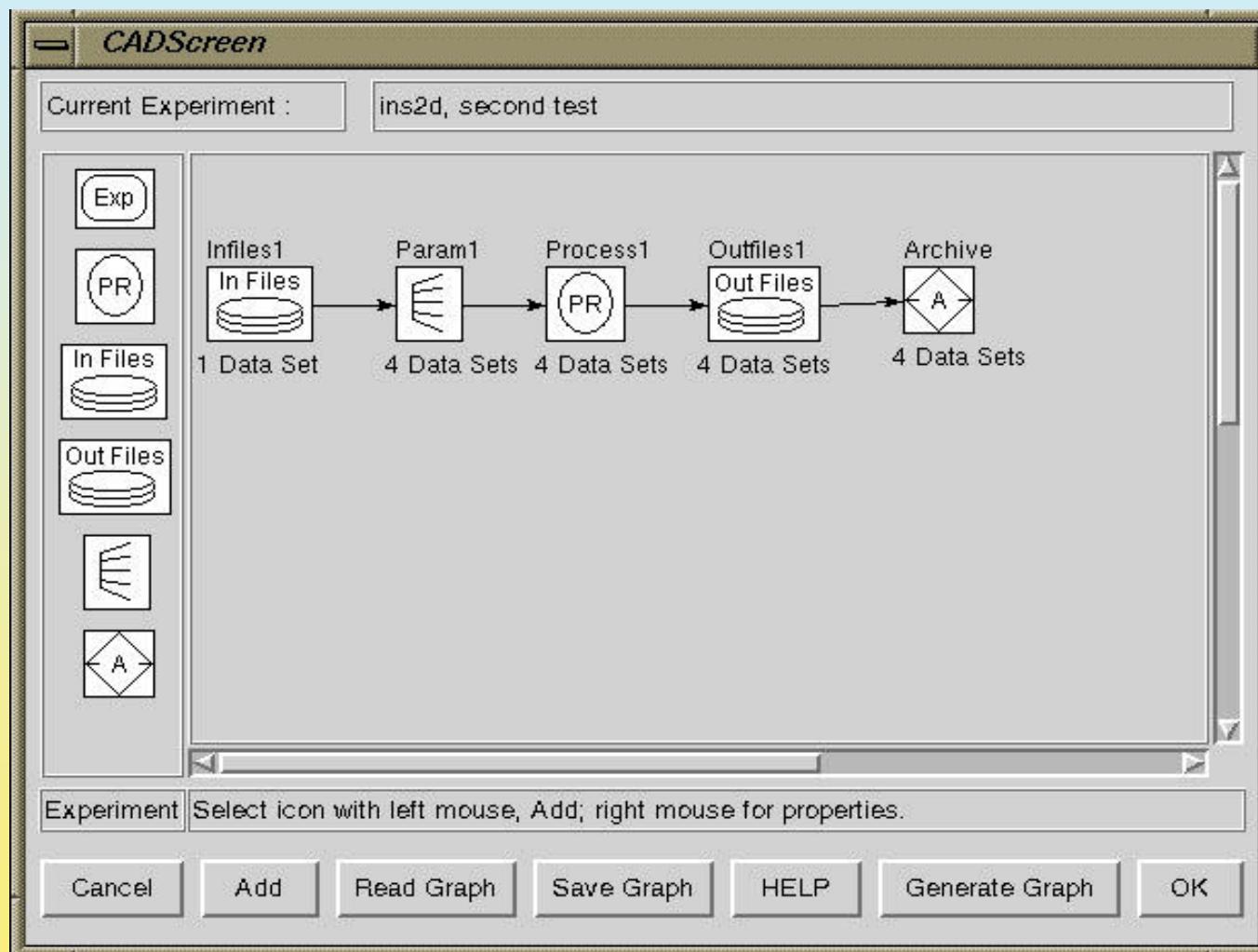
X38 Crew Return Vehicle

- 192 cases (16 X 12 parameter study)
- full 3-D Navier-Stokes (Overflow-d2)
- 128 grids, 2.5 million grid points





ILab CAD SCREEN FOR PROCESS SPECIFICATION





FUTURE DIRECTIONS

**Implement CAD screen experiment creation
for complex process specification
(in addition to current wizard screens)**

- visual scripting

Additional Job Models

- Condor, Legion, PVM ?

**Implement plug-in's for different types
of monitoring and scheduling**